MAINTAINING AND USING PROVENANCE DATA IN COMPUTER MUSIC

John ffitch
Codemist Ltd
Combe Down, Bath

Julian Padget
University of Bath
Department of Computer Science

ABSTRACT

E-Science is increasingly recognising the importance of provenance, that is the origin and subsequent history of a document, in assessing its value. We argue here that computer music can also benefit from incorporating indications of provenance at many levels. We recognise that the challenge is to create and maintain metadata non-invasively ... until, that is, the composer or musicologist wants access. Although this paper, having explored the general idea of computational provenance, describes an application to composition, in particular using the Csound system, the same idea could be incorporated into other systems. We also look forward to further possible applications.

1. INTRODUCTION

Surely we all have done it! Deep in the process of composing a piece we adjust details of the timbres or timing, moving sections a little or adding a new layer. Then comparing the new with the old we decide that the last change was a mistake, and we wish to revert, only to find that in the excitement of the moment, or possibly because the man from Porlock called, [1], we cannot remember what we changed or how to return to the previous state. We may have the audio file, but not the program or process that created it.

Apart from telling ourselves that next time we will keep better notes, what can we do?

We are arguing in this paper that the solution lies in a special case of the more general concept of **provenance**. This notion is usually associated with the area of artistic artifacts, who owned them and how they reached the saleroom. But the idea can be of use in a much more local way. We will describe an initial system for this in the context of Csound[2] compositions, and consider the drawbacks, extensions and future needs.

2. PROVENANCE ON THE WEB

In e-Science there have been a number of provenance projects, aimed at reproducibility of experiments and implemented by recording the operations which have been carried out, when and by whom. For example in the Pasoa project[5] it was said

> The importance of understanding the process by which a result was generated in an experiment is fundamental to science. Without such information, other scientists cannot replicate, validate, or duplicate an experiment. We define provenance as the process that led to a result.

It is easy to see that the same is true when the result is a musical composition, even if the composer keeps track of the process entirely within his own head. Here we are going to adapt this general concept for the particular case of composition of computer music.

3. PROVENANCE IN COMPOSITION

Musicologists are already interested in the provenance of different versions of computer pieces, because they want to know about the evolution of the composer's thoughts. For example Nuhn *et al.*[8], Burns[1] and ffitch *et al.*[3], have spied on composers and analysed the remnants of the process of composition in the form of paper and audio files. This area would be greatly enhanced by the incorporation of a provenance system.

In this paper we are concerned with an initial, weak form of recording provenance, which we think will be of particular interest to composers and might also eventually be of use in musicology. We wish to start simply by associating a section of audio with the process that created it.

What we are seeking is to incorporate within the output file all the necessary information to recreate that file, together with sufficient tools to recover or inspect that information. In the next section we describe an initial attempt at such a system.

Of course this is not totally a new idea; indeed the Desktop Composer's Project[4] long ago made available additional space in its sound file format header where the composers could record any information they wished, and this was taken forward in the IRCAM format where the header block is larger than required for the usual sample rate, number of channels and the like. It would have been possible, for example, to record here the command line and arguments in the creation of the output. Because CDP

[1] In case it is needed `http://en.wikipedia.org/wiki/Person_from_Porlock`

was largely a collection of transformations of an input audio file to an output audio file, by also copying the provenance data from the input to the output a complete history of the creation could have resulted. The problem is that this required user action, while everyone concerned with metadata knows that the trick is to avoid human authoring and implement mechanisms for automated capture (and validation) of the information. As we said in the abstract, the challenge is going to be how to create and maintain metadata non-invasively, while making it easily available when required.

4. INCORPORATING IN CSOUND

Already within Csound5 it is possible to add the conventional information of a licence, title, copyright, artist, date and other comments, using the mechanisms of *libsndfile* and the `sf_set_string` function. This is a minimal scheme and does not fulfil our requirements. However it does give us a hint.

If we look at file formats and systems for other kinds of data, such as JPEG for pictures and album software like Picasa, music file formats like MP3, OGG and FLAC, or video archives such as DVDs it is clear there are two solutions, each with their own pros and cons: embed the metadata or keep it in a parallel associated file. Separation means all existing tools continue to work, but there are significant difficulties in ensuring consistency and integrity of the metadata. Embedding impacts the tools that are used to process the data since they have to be capable of ignoring elements they are not programmed to handle. The preferred scenario is that metadata capacity is built into the data format from the beginning, since retro-fitting essentially mandates the separation approach if backward compatibility is to be maintained or a painful transition is to be avoided. We now discuss these issues in the context of WAV files and Csound.

4.1. A simple scheme

There are two fundamentally different ways to achieve our aim. The first, the one we have not implemented, is to make Csound generate an XML file as output, incorporating the audio file in the schema. The disadvantage of this approach is that we would need to write an audio player in association, and probably plugins for Audacity and similar software to use the output.

The method we did use was to exploit the definition of the RIFF format, defining new chunk types to record the score, orchestra and command line. The definition of WAV format[6] says that if a processor encounters a chunk name that it does not recognise then it should ignore and skip it. This means that while further processing may not preserve the information, the process will not stall, and we will be able to review the audio, listen, or display; even transfer to CD.

The initial experimental system was implemented as a small C program that remembers the command line and from that the orchestra and score, or the unified CSD file. It then calls Csound via a *system* call, and finally adds the new chunks to the output file identified from the command line. The names of these basic new chunks are given in table 1. We would have preferred to use the generic *libsoundfile* library to handle these additional chunks, and we still plan to do that in a later version. This scheme is limited to WAV format at present but should be easy to extend to AIFF.

Chunk ID value	Purpose
CARG	Command line arguments
CORC	Orchestra file
CSCO	Score file

Table 1. Primary Additional Chunks

Chunk ID value	Purpose
CIFN	Input file name
CEFN	Extract file name acting on score.srt
CMDF	name of MIDI event file

Table 2. Extra Additional Chunks

Of course just adding the information to the audio file output is not all that is required. It is also necessary to recover the information. To this end we created a small utility that could scan the output file and display the chunk names or the data or extract the information to files. It is important that both inspection and recovery exist as we may not wish to overwrite a later version.

4.2. Additional Issues

The system outlined above is sufficient for simple compositions that use entirely synthetic sounds. However many composers work with samples, either recorded or created by other means. This means that the orchestra score and command line does not include the entire material. Similarly the score might be expressed as a MIDI file rather than a Csound event-list. Particular variants in the case of Csound include reading a soundfile as input (`-i` command line option) or using an extract of a previous score (`-x` option). These are included in our simple system with chunk names described in table 2.

5. USE OF THE PROVENANCE SYSTEM

The way in which the composer uses this system is that he initially uses our version of Csound rather than the standard one. This is in effect only a wrapper, and so provides all the same options and language as the real system. At present the effect is limited to WAV file output; any other format, including real-time, are totally unaffected. The composer creates the audio file, listens using a standard audio player like `aplay` or a sound editor like `audacity`. As a result of the audition the score and/or the orchestra is adapted and the process repeated. As long

as the composer synthesises the audio to a different file on each occasion, (s)he is able to listen to a number of versions and decide which is preferred. At that stage the small utilities can be run to look at the differences between the input programs, and either new revisions or merged versions can be created.

If our composer is organised, and keeps notes throughout the process, then the cost of the scheme is minimal and it may never be used. If however the notes are incomplete, or enthusiasm overtakes process, then there is an insurance scheme in place, and there is a route to recovery, or even to subsequent careful analysis.

6. EXTENSIONS AND THE FUTURE

So far we have described a basic system. In this section we consider how it needs to be extended and ponder alternative approaches.

6.1. What is Missing

The first and most obvious problem with the simple scheme is what to do about audio samples, that are typically stored in tables in Csound. The simpler issue here is noticing them. That means either a static system of scanning the score for file names in ftables, or more reliably a dynamic system that traps all loading of files and records the file name. This also serves to highlight the drawback of the simple wrapper. If we are to record all uses of external data we need to police certain opcodes, including those that create ftables, which means we ought to have a more integrated system, where the opening of files for reading can be intercepted and treated appropriately.

A further problem with samples is whether to record the name of the file containing the sample, or the sample itself. In most cases we may assume that the samples are static and unchanging, in which case it would be normal to record just the name. In addition the adding of the samples would increase the size of the output audio. The counter argument is that recording only the names will not provide security again the wholesale deletion of all samples [2] . We have no immediate solution for this. We are considering an extension of the file name regime in Csound, so we save file data and record URL-style data as names.

There is also a worrying dependence on generating WAV audio files, ignoring both other formats. Clearly there are other audio formats that are RIFF-based and can accept additional chunks, but for the others, and critically for real-time output, something different is required. The easiest solution would seem to be to create a separate file of the date, almost a fake WAV file, with a name including a time stamp. This is not in keeping with our original goal of safe and unavoidable protection, but is the only way we have identified so far.

A similar issue arises if a piece uses real-time MIDI input, or the similar real-time event stream. In the case of MIDI input it should be possible to capture the events and create a MIDI file from them, and save this. Similarly if the event stream is created by an external process, then it needs to be saved and created as a file of events.

[2] Yes, one of us has done that, being left with the audio, the orchestra and score, but not the samples.

6.2. Alternatives

Before leaving the collection of new features that are required it is worth considering whether there are alternative methods. One such idea would be to insist on a transactional filing system, with the possibility of rolling back the changes. Over many years there have been experiments with database filing systems (for example we used the relational database extension to the TRIPOS system in the 1980s[9], and PICK[11] was in commercial use) but they have not reached widespread use. There are still problems with exactly how such a system might be packaged for natural use in a provenance system rather than in content-addressable data, but this may have to be be a long term project.

A similar scheme would be to make use of a filing system that performs snapshots such as SUN's ZFS[10] or Microsoft's NTFS[7]. These systems can be arranged to record deltas (changes) at fixed intervals. The problems of how often to take the snapshot and what to do to recover files make these solutions less than satisfactory.

Many programmers would assert that the obvious technology is a software repository such as CVS or SVN. Currently these systems require explicit user actions, and so fail our transparency rule. It would be possible to consider a wrapper that did checkins as part of the synthesis, but there are longer term problems of persistence and remembering too much. However we would expect this technology to be used at some stage.

7. CONCLUSIONS

It has become apparent that what started as a simple safeguard to stop one of us losing his files has grown into a complicated activity full of possible alternatives. Our initial system is sufficient for pure computer synthetic composition but is less satisfactory when dealing with sample and external data. The problems of static versus mutable data remain currently unanswered.

However we have shown that it is possible to add a provenance system to an existing synthesis system with no disruption to the working practices of the user. We would like to see other systems adopting mechanisms whereby provenance data is recorded, both for composer protection and for subsequent investigations. We have shown that a simple system is both easy to create and useful in practice.

We will be considering making the Csound provenance system available in the next months, and we will be seeking suitable compromises to deal with the remaining problems, while maintaining the transparency that will make it acceptable to users.

We would like to express our thanks to members of the University of Bath Media Technology Centre for dis-

cussions and criticisms, and Archer Endrich in particular. Some initial coding was done by Sayeh Iliafar.

8. REFERENCES

[1] BURNS, C. Tracing Compositional Process: Software Synthesis Code as Documentary Evidence. In *Proceedings of International Conference on Music and Computers (ICMC)* (Göteborg, Sweden, Sept 2002), pp. 568–571.

[2] CABRERA, A., VERCOE, B., FFITCH, J., PICHÉ, J., NIX, P., BOULANGER, R., EKMAN, R., BOOTHE, D., CONDER, K., YI, S., GOGINS, M., PINOT, F., AND KOZAR, A. *The Canonical Csound Reference Manual*, 5.08 ed. `http://sourceforge.net/projects/csound/`, Feb 2008.

[3] DOBSON, R., FFITCH, J., LOMBARDO, V., TAZELAAR, K., AND VALLE, A. Varèse's *Poème Électronique* Regained: Evidence from the VEP Project. In *ICMC 2005 free sound* (2005), SuviSoft Oy Ltd, Tampere, Finland, Ed., Escola Superior de Música de Catalunya, pp. 29–36. `http://www.cs.bath.ac.uk/~jpff/PAPERS/DobsonffitchTazelaar05.pdf`.

[4] ENDRICH, A. Composers' Desktop Project : a musical imperative. *Organised Sound 2*, 1 (1997), 29–33.

[5] GROTH, P., MILES, S., AND MOREAU, L. Preserv: Provenance recording for services. In *Proceedings of the UK OST e-Science second All Hands Meeting 2005 (AHM'05)* (Nottingham,UK, Sept. 2005).

[6] IBM, AND MICROSOFT. Multimedia programming interface and data specifications 1.0. available at `http://www.kk.iij4u.or.jp/~kondo/wave/mpidata.txt`, August 191.

[7] MICROSOFT. How NTFS Works. `http://technet2.microsoft.com/windowsserver/en/library/8cc5891d-bf8e-4164-862d-dac5418c59481033.mspx?mfr=true`, March 2003.

[8] NUHN, R., EAGLESTONE, B., FORD, N., MOORE, A., AND BROWN, G. A qualitative analysis of composers at work. In *Proceedings of International Conference on Music and Computers (ICMC)* (Göteborg, Sweden, Sept 2002), pp. 572–580.

[9] RICHARDS, M., AYLWARD, A. R., BOND, P., EVANS, R. D., AND KNIGHT, B. J. TRIPOS – a portable operating system for mini-computers. *Software Practice and Experience 9* (1979), 513–526.

[10] SUN. ZFS: the last word in file systems. `http://www.sun.com/2004-0914/feature/`, 2004.

[11] WYATT, M., AND BATE, J. *The PICK Operating System*. Blackwell Science (UK), 1986.